CIT 591 Introduction to Software Development

Module 7: Software Testing

Module Learning Objectives

- Understand the practice of test driven development
- Write code such that it is easier to test
- Create Junit tests within the Eclipse framework
- Measure the usefulness of a test in terms of number of lines of code covered

Module Glossary

- **Unit testing:** A level of software testing where individual units are tested. The hope is that once the individual pieces are tested and verified to be working, then the likelihood of the overall project working becomes much higher. For Java programs, a unit is usually a method.
- **Test driven development (TDD)**: A programming practice in which tests are written before actual code is written. The goal of the programmer then becomes that of ensuring that they write code such that all the tests pass.
- **Assert statement:** A statement wherein the developer is asserting that something needs to be true. Usually, if the assertion fails, code execution is halted. In the context of JUnit testing, built-in assertion mechanisms are provided by the Assertions class. Official documentation
- **`assertEquals`:** An inbuilt assertion used for asserting that the actual value of a variable (or the value returned by invoking a method) matches the expected value. Most unit tests will use this statement extensively.
- **Test coverage:** When executing the set of tests, if any line of code is executed, that line is said to be covered. Test coverage is used in Software Testing to quantify the amount of code that is being covered by a set of tests, typically provided as a percentage of lines covered.

Module Resources

- **Textbook Readings:**
  - Unit Testing
    - 3.4: Unit Testing
    - 8.7: Unit Testing Frameworks

    - Test Coverage
        - 5.6: Problem Solving: Selecting Test Cases
- **Websites:**
    - JUnit
    - JUnit Assertions
    - Changing your Eclipse color scheme
    - Installing EclEmma for Eclipse
    - EclEmma - Launching In Coverage Mode
    - Lines of Code for Real World Systems
    - The Staggering Impact of IT Systems Gone Wrong
    - Why Software Fails
    - TDD is Dead, Long Live Testing
      Note: David Heinemeier Hansson (DHH) is creator of Rails. While this is a different programming language, the sentiment expressed in this note by him are noteworthy even to Java developers.

Key Concepts & Examples

JUnit testing:
Writing the tests before you write all of the actual code is called Test Driven Development (TDD). Eclipse makes creation of these tests easy using JUnit tests. Use assert statements in your unit tests to ensure that the actual result matches your expected result.

- Examples used in the video: testing code that adds fractions

Test coverage:
Test coverage is a measure of the percentage of lines of your actual code that are being hit by your tests. Use a tool like EclEmma to measure this. General practice is to strive for at least 80 percent coverage (higher is obviously better), but remember that 100 percent coverage does not mean that you have handled all possible test cases. It is not uncommon to have more than one test to cover a given unit e.g. it is possible, and is often recommended, to have tests that account for both expected and unexpected behaviour of a given unit.

- Examples used in the video: Test coverage using a BankAccount class